Resource Configuration and Allocation

A Case Study of Constrained Heuristic Search

Neena Sathi, Mark S. Fox, Rajay Goyal, and Alexander S. Kott
Carnegie Group

CORAL IS A REAL-TIME, INTERACTIVE DECISION-SUPPORT SYSTEM THAT HELPS INVENTORY MANAGERS CONFIGURE AND ALLOCATE COMPONENTS TO CUSTOMER ORDERS. USING CONSTRAINED HEURISTIC SEARCH, CORAL PRODUCES NEAR-OPTIMAL SOLUTIONS QUICKLY.

WHEN CUSTOMERS ORDER specific quantities of products with particular features or components, inventory managers must perform two tasks: first they design feasible product configurations of component parts based on physical compatibilities and user-specified preferences, and then they try to allocate optimal quantities of components to those specified configurations based on available inventory. If a component can be used across multiple configurations, supplying one configuration can affect an inventory manager's ability to supply other configurations. The manager must project how many orders can be configured from an inventory and which components will need to be ordered—a time-consuming job.

These tasks are difficult because feasible product configurations are limited by interactive constraints (electrical, thermal, geometric, and so on), and because limited inventory and alternative configurations increase the complexity of allocating resources. Alternative configurations represent alternative ways a single product can be supplied, and an order for n products can be and often is satisified by more than one configuration.

We have applied constrained heuristic search techniques[1] to the problems of product configuration and inventory planning. Our interactive, real-time decision support system, called Coral, maximizes the total number of products requested, given one or more substitutable configurations. This gives inventory managers a tool for managing and allocating component inventories and thus maximizing the number of complete orders. Coral is also intended to perform "what-if" analyses on possible order demands. This approach views configuration and allocation as constraint-satisfaction and optimization problems, where constraints guide the problem solving (that is, searching). Spex, a problem-independent version of the configuration module, has been developed and applied to other configuration problems.

Problem definition: Planning ammunition rounds

Configuring complete ammunition rounds and allocating inventory can be a problem for army logistics.[2] A complete round of ammunition contains projectiles, cartridge cases, propelling charges, primers, and fuses. An army unit's mission and environment dictate the requirements for a complete round. For example, an area with reduced visibility might require Illum (illumination) rounds consisting of Illum projectiles fitted with time fuses. Similarly, material targets require HE (high-energy) projectiles. For maximum tactical flexibility, planners can use different components across different types of rounds. For example,

0885/9000/92/0400-0026 $3.00 © 1992 IEEE

for a 155-mm towed Howitzer artillery, an HE projectile with a normal cavity can be fitted with any of six types of fuses: three PD (point-detonating) models, two MTSQ (mechanical-time superquick) models, or a Prox (proximity) model. Similarly, one kind of PD fuse can be used across various projectile types (Agent, HE, or Hera) for the same Howitzer. Thus, the same components can be used to configure different types of rounds. This interchangeability increases the difficulty of determining the impact of one type of round on inventory and thereby its effect on the availability of other types of rounds.

The configuration/allocation problem is made more difficult, from a decision-support perspective, by the way that problems can be posed. The following are common planning problems in field situations:

- Maximize the number of 155-mm HE, Illum, and Smoke rounds available at ATP-1 (the ammunition transfer point, where ammunition from the rear is transferred to the front), using any combination of compatible components.
- Configure the maximum number of 105-mm Smoke rounds with 20 percent PD fuses, 50 percent Prox fuses, and 30 percent MTSQ fuses.
- Configure the maximum number of 155-mm HE, 105-mm Smoke, and 105-mm Illum rounds from the inventory at ATP-1 and ATP-2 storage locations. If different rounds compete for components, maintain the following proportion: 40 percent HE rounds, 30 percent Smoke rounds, and 30 percent Illum rounds.
- Configure the maximum number of rounds from the inventory at ATP-3. If different rounds compete for components, maintain the following proportions:
  - 40 percent 155-mm HE rounds: 20 percent use PD fuses, 50 percent use Prox fuses, and 30 percent use MTSQ fuses; half use percussion primers and half use electric-percussion primers;
  - 30 percent 105-mm Smoke rounds, with 10 percent CP fuses, 40 percent Delay fuses, and 50 percent Prox fuses; and
  - 30 percent 105-mm Illum rounds with a default mix of fuse, primer, and propelling-charge components.

Problem: Maximize total number of ammunition rounds by suggesting feasible configurations of each round (with specific components to use in each configuration) and how many of each configuration to use.

Parts: Ammunition components

Inputs:
- Inventory map $I_i(A_i, Q_i)$ for $(i=1,2...n)$, where
  - A_i — Ammunition component model
  - Q_i — Ammunition component quantity
- Description of round $R_i(N_i, C_i, P_i, F_i, PC_i, Pr_i)$ for $(i=1,2...n)$
  - N_i — Composition of the round
  - C_i — Cannon size or model number of the weapon
  - P_i — Projectile or cartridge type (by function only)
  - F_i — Fuse types and compositions
  - PC_i — Propelling-charge types and compositions
  - PR_i — Primer types and compositions

Constraints:
- Maximum inventory for each ammunition component
- Quantity or composition constraint on round or component
- Ammunition compatibility constraints

Outputs:
- Legal configurations
- Assignment and allocation of component inventory to feasible configurations
- Analysis of planning results

Criteria of evaluation:
- Maximum use of storage inventory

Figure 1. Problem description for ammunition round planning.

The configuration and allocation task is further complicated by the need to explore alternative requirements. For example, the planner might want to see how well the same set of goals can be accomplished with different inventories, or compare how well a single inventory can accomplish several different sets of goals. By making small changes in the goals, the planner can perform "what-if" analysis—for example, determining how many 155-mm HE rounds can be configured if the target for 155-mm Apers (antipersonnel) rounds is increased. The planner might also want to generate unrelated plans for various sets of storage locations, each with its own set of goals. Figure 1 describes inputs, outputs, and constraints for the ammunition-round planning problem.

Planners usually view inventory from the component level rather than from the configuration level, so it is difficult for them to project which components need to be ordered and how the components in inventory can be put together into complete rounds that are likely to be needed. It is time consuming to play what-if games on the possible availability of ammunition rounds and to answer questions like these:

- What is the maximum number of different rounds in the inventory?
- What are the relative numbers of rounds of different types present?
- Which part numbers can be configured together in a single round type?
- What will be the inventory shortfall if certain types of rounds with a given composition need to be configured?

Constrained heuristic search

Previous research has focused on solving either the configuration or the resource allocation problem, each independent of the other (see the sidebar on p. 28). In Coral, we view both as instances of constrained heuristic search. CHS is a problem-solving method that formalizes the concept of problem structure and its analysis, and provides powerful heuristic advice to guide search. In particular, the model augments the definition of a problem space

Previous research

Configuration planning has been approached using pattern-directed inference[1] and problem decomposition combined with constraint propagation.[2,3]

Recently applications of AI to design configuration follow one of two approaches. The decompositional[2,4-6] (or abstract refinement[7]) model represents the configuration process as a sequence of hierarchical refinements. Each successive refinement produces a more detailed description of the artifact, without altering its initial structure. The transformational model[8-10] represents the design configuration process as a sequence of restructuring steps. Each successive transformation replaces a part of the design structure (for example, a group of components and their connections) with a different substructure, but essentially at the same level of detail.[11] Example configuration problems include the configuration of manufacturing equipment, manufacturing systems, group technology cells, part groups, orders, shipments, and products.

Coral's approach to configuration is most related to the DSPL implementation of an air cylinder design system.[4] Both approaches handle design problem solving through hierarchical decomposition, contain specialists that guide design at each level, and use constraints as a means of determining admissible design decisions.

On the other hand, the DSPL approach encodes the routine design process by hierarchically decomposing the artifact's function. Coral takes a structural approach, decomposing the artifact (family) itself, and represents all possible combinations of the artifact in a configuration tree. Also, DSPL admits arbitrary problem-solving methods at each node in the tree, whereas Coral relies primarily on constraint knowledge to determine the admissability of design decisions. Coral is more limited in the design problems it can tackle, but by limiting the set, the design knowledge becomes more declarative and easier to encode.

Example resource-allocation problems include the allocation of workstations among engineering managers; classrooms among professors or department administrators; troops, artillery, or hardware in a battle situation; airport gates to incoming and outgoing flights; and so on. In a typical resource-allocation situation, there are sets of agents, each with a set of allocated resources employed against a set of activities requiring resources. Some researchers have focused on the resource allocation and reallocation aspect rather than on configuration.[12,13] Others have applied mathematical programming techniques to resource allocation by treating it as an assignment problem with linear constraints.[14]

References

1. J. McDermott, "R1: A Rule-Based Configurer of Computer Systems," *Artificial Intelligence*, Vol. 19, No. 1, 1982, pp. 39-68.
2. S.M. Mittal, C.L. Dym, and M. Morjaria, "Pride: An Expert System for the Design of Paper-Handling Systems," *Computer*, Vol. 19, No. 7, July 1986, pp. 102-114.
3. S. Marcus, J. Stout, and J. McDermott, "VT: An Expert Elevator Designer That Uses Knowledge-Based Backtracking," *AI Magazine*, Vol. 9, No. 1, Spring 1988, pp. 95-114.
4. D.C. Brown and B. Chandrasekaran, "An Approach to Expert Systems for Mechanical Design," *Proc. IEEE CS Trends and Applications Conf.*, 1983, pp. 173-180.
5. M.L. Maher, *Hi-Rise: A Knowledge-Based Expert System for the Preliminary Structural Design of High-Rise Buildings*, doctoral dissertation, Carnegie Mellon Univ., Pittsburgh, 1984.
6. K. Preiss, "Data Frame Model for Engineering Design Process," *Design Studies*, Vol.1, No. 4, 1980, pp. 231-243.
7. J. Mostow, "Toward Better Models of the Design Process," *AI Magazine*, Vol. 1, No. 1, Spring 1985, pp. 44-57.
8. A. Howe et al., "Dominic: A Domain-Independent Program for Mechanical-Engineering Design," in *Applications of Artificial Intelligence in Engineering Problems, Vol. 2*, D. Sriram and R. Adey, eds., Springer-Verlag, New York, 1986, pp. 289-300.
9. R. Joobbani, *Artificial Intelligence Approach to VLSI Routing*, Kluwer Academic Publishers, Hingham, Mass., 1986.
10. S.S. Murthy and S. Addanki, "Prompt: An Innovative Design Tool," *Proc. Sixth Nat'l Conf. Artificial Intelligence (AAAI '87)*, MIT Press, Cambridge, Mass., 1987, pp. 637-642.
11. A.S. Kott and J.H. May, "Decomposition Versus Transformation: Case Studies of Two Models of the Design Process," *Proc. 1989 ASME Computers in Eng. Conf.*, Assoc. of Mechanical Engineers, New York, Vol. 1, 1989, pp. 1-8.
12. A. Sathi, *Cooperation through Constraint-Directed Negotiation: Study of Resource Reallocation Problems*, doctoral dissertation, Carnegie Mellon Univ., Pittsburgh, 1988.
13. R. Camden et al., "Distribution Planning: An Integration of Constraint Satisfaction and Heuristic Search Techniques," *Proc. Logistics Directorate (J-4) Joint Staff, Symp. on Artificial Intelligence Applications for Military Logistics*, Am. Defense Preparedness Assoc., Arlington, Va., 1990, pp. 177-182.
14. V. Srinivason and A.D. Shocker, "Linear Programming for Multidimensional Analysis o f Preferences," *Psychometrika*, Vol. 38, 1973, pp. 337-369.

(composed of states, operators, and an evaluation function) by refining a state to include

(1) the problem topology, or structural characterization;
(2) problem textures, which measure a problem topology to focus search and reduce backtracking; and
(3) the problem objective, a means for rating alternative solutions.

A problem's topology is defined by a constraint graph in which nodes are variables and arcs are constraints. Problem objectives are embedded in the topology as utilities (that is, a number between 0 and 1 that reflects how well the value optimizes the objective) associated with values in a variable's domain. Propagation across constraints within a topology (also known in the constraint-satisfaction literature as arc consistency) reduces the domain of variables and alters the utilities associated with the remaining values. After propagation, a value's utility is no longer locally defined but reflects its global interaction, via constraints, with other variables.

Search operators in constrained heuristic search have many roles. They refine problems by adding new variables and constraints, reduce the number of solutions by reducing the domain of variables (for example, by assigning a value to a variable), and reformulate problems by relaxing or omitting constraints or variables.

For constrained heuristic search to be well focused, we must decide where to apply an operator in the problem topology. Features of the topology must exist that differentiate one subgraph from another, and these features must be related to the problem's goals. Seven features, called problem textures, have been identified and are being analyzed.[3]

Given these definitions of problem topology and textures, the problem-solving model begins the search process with a single state that includes the initial problem topology. The model propagates constraints through the graph, computes texture measures, and selects a decision node (described in more detail below). The system then generates a new state by selecting an operator that either adds structure to the topology or further restricts a variable's domain at the decision node. Thus, through the search process, the constraint graph is successively transformed into a solution.

With this approach we can construct a nearly optimal solution more quickly, manipulate the goals interactively, and gauge the impact of changes.

Coral architecture

Coral's architecture has three major modules, shown in Figure 2: a configurator, a resource allocator, and a plan analysis module.

The configurator creates all feasible configurations of a product based on compatibility constraints. These configurations are then given to the resource allocator, which assigns quantities to each feasible configuration based on component inventory and quantity and composition constraints. The analysis module identifies bottleneck components and analyzes user-specified constraints on composition. For example, if a user-specified composition can be relaxed, the module can determine how many more orders can be filled.

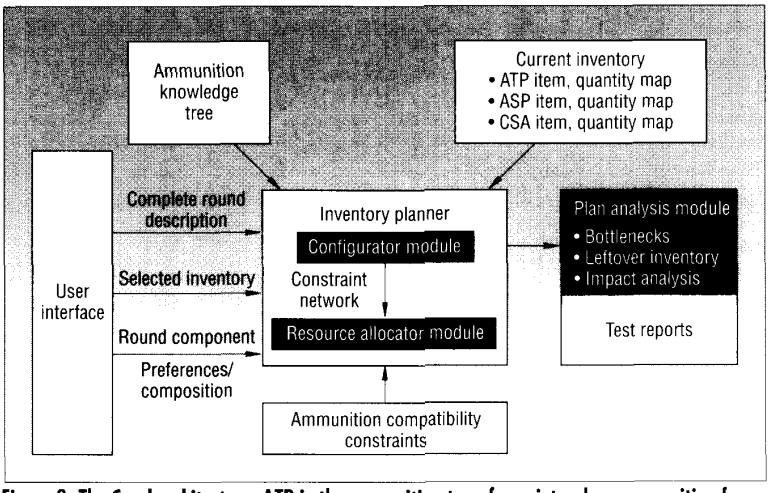


Figure 2. The Coral architecture. ATP is the ammunition transfer point, where ammunition from the rear is transferred to the front. ASP is an ammunition storage point, which supplies an ATP. CSA is a corps storage area, which supplies an ASP.

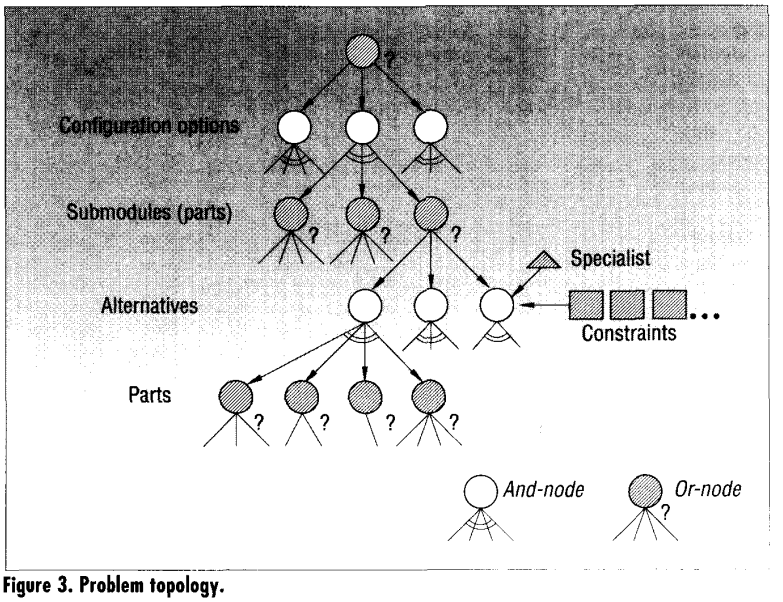


Figure 3. Problem topology.

The configurator. We use a technique that is well suited for configuring "decomposable" artifacts with reasonably well defined structures and constraints.[4] This methodology is one of the decompositional approaches to configuration, and is intended for a weakly connected, "nearly decomposable" configuration artifact.[5,6] Such an artifact can be subdivided into parts or characteristics with relatively weak interactions. Some of these parts or characteristics can, in turn, be decomposed into subcharacteristics or parts, and so on. Similarly, the task of configuring an order can be subdivided into the tasks of configuring its major modules (or major features); the task of configuring a major module can be decomposed into tasks of configuring its major components or subfeatures; and so on, until we reduce the problem to the task of choosing between standard arrangements or parts.

The decompositional model represents the configuration process as a sequence of refinements, each of which starts with an incomplete configuration state and produces a configuration state of a greater completeness (in the sense that the new state contains more information about the configuration object). It does this by adding to one of the components its more detailed description (either its specific "committed" implementation, or its decompositional description). The new configuration state's structure is the same as the initial state's structure, at least at the level of abstraction found in the initial state.

Problem topology. Decomposition treats the artifact as a tree-like structure. The root of the tree corresponds to the final configuration artifact. The leaves are elementary objects, which are either predefined or are simple enough to be configured by predefined procedures. In general, each part of the tree can be configured in more than one way and, correspondingly, can have more than one decomposition. Hence, the problem topology is represented by an And-Or tree, as shown in Figure 3, and is known before the configuration process begins. In a decomposable configuration problem, all

possible configurations of the artifact are implicitly defined by the problem topology (also known as the configuration knowledge tree). However, the space of all possible configurations is usually very large, and finding a configuration that satisfies a certain set of requirements (constraints and goals) is a computationally explosive problem. (It is an example of a disjunctive constraint graph,[7] which can also be represented as a dynamic constraint graph.[8])

The topology is traversed hierarchically from top to bottom; higher level disjunctive decisions activate lower level variables and constraints. Constraint checking takes place only among active variables. Each constraint acts on one or more components, either by checking that the constrained component satisfies a certain condition encoded in the constraint, or by verifying proper relations between two or more components (for example, their compatibility). The constraint's primary role is to evaluate the choice of values for the decision variables. This procedure accepts the values of one or more decision variables and returns the utility associated with this combination of values.

Operators. Operators use domain-specific knowledge to guide their search through the space of possible configurations. They are expected to guess which value of the decision variable should be tried first when constraints do not have enough information to provide the answer, or to guess which variable should be assigned next when there are no fully constrained variables. Because there are two kinds of nodes (And-nodes and Or-nodes), there are two kinds of operators, And-specialists and Or-specialists.

And-specialists choose the sequence in which the components defined in an And-node are to be instantiated. Depending on the active goals and constraints for a given configuration session, the sequence can vary, leading to different configurations. In choosing a sequence, an And-specialist refers to previous decisions, the constraints relevant to its node, and the configuration goals. To order the And-node parts, Coral enters all this information into the And-specialist's rule or procedure and then processes it.

Or-specialists select the most appropriate alternative among several that are associated with their respective Or-node. Each Or-specialist contains procedures or rules that consider previous decisions, constraints on its node, and configuration goals when making a selection. These procedures and rules can be used to

- rank all available alternatives and pick the one with the highest ranking,
- name a single alternative that is suitable under current conditions, or
- generate an alternative (for example, by looking up a parts list).

SINCE CORAL RECORDS EACH DECISION AND ITS ALTERNATIVES, THE SYSTEM CAN ANSWER "WHAT-IF" QUESTIONS AFTER THE USER CHANGES CONSTRAINT GRAPH PARAMETERS.

Search. The process of defining a valid configuration is a search for a consistent subtree of the configuration knowledge tree, where each Or-node is assigned a single alternative. The initial state is composed of the problem topology, that is, the configuration knowledge tree.

(1) The configurator begins the search by selecting one of the states (a partial configuration). When a single optimum configuration is sought, it is beneficial to select a state in a "best-first" fashion. If all feasible configurations need to be defined, the selection can be made at random.
(2) The local And-specialists rank the children of And-nodes, thereby selecting the branch where the next configuration decision should be made. The first unresolved Or-node found is the decision node.
(3) The local Or-specialist then selects and assigns the most promising alternative to the decision Or-node. If all alternatives have been exhausted, the partial configuration is deleted and the process returns to step 1.
(4) The configurator adds the new assignment to the partial alternative, creating a new state with an extended partial configuration. The module identifies and checks all constraints relevant to the last assignment. A constraint is potentially relevant if it is related either to the decision node itself or to any of its ancestors (either an And-node or an Or-node). If the new partial configuration passes all the constraints, it is added to the list of partial configurations, and the process iterates from step 1.
(5) If the partial configuration violates a constraint, the configurator can reconsider previous decisions through dependency-based and knowledge-based backtracking, since a specific constraint can be associated with information about ways to fix that violation.[8,9]
(6) Search ends when a feasible subtree is found.

The resource allocator. For each requested product, the configurator provides the resource allocator with one or more configurations that satisfy the product requirements. Each acceptable product configuration requires the conjunctive allocation of one or more of each component. Maximizing the total number of product configurations depends on the allocator's ability to allocate bottleneck components. We can optimize the solution to this problem using constrained heuristic search.

Problem topology. The problem topology is composed of nodes representing user-defined product specifications, acceptable product configurations generated by the configurator, and components. Links between nodes represent demand.

For the problem of planning complete ammunition rounds, let's assume the user has specified the following composition:

- $Round_1$ (155-mm HE): 40 percent
- $Round_n$ (105-mm Apers): 60 percent

For $Round_1$, the user wants to maintain this component composition:

- Projectiles: 100 percent
- PD fuses: 60 percent
- MTSQ fuses: 40 percent

For $Round_n$, the user requests 20 percent more fuses than projectiles, and wants

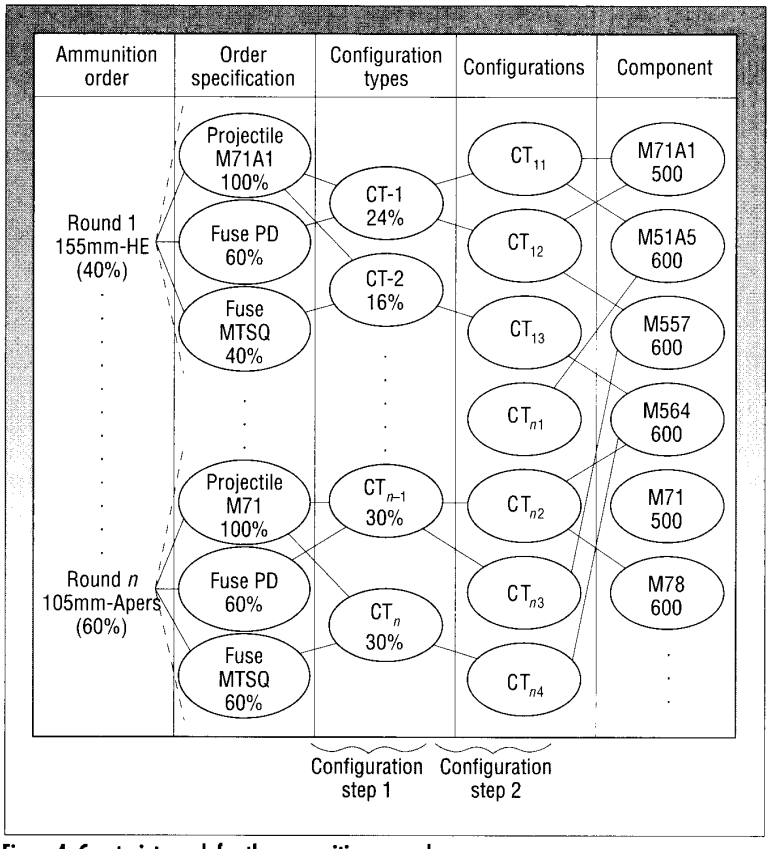


Figure 4. Constraint graph for the ammunition example.

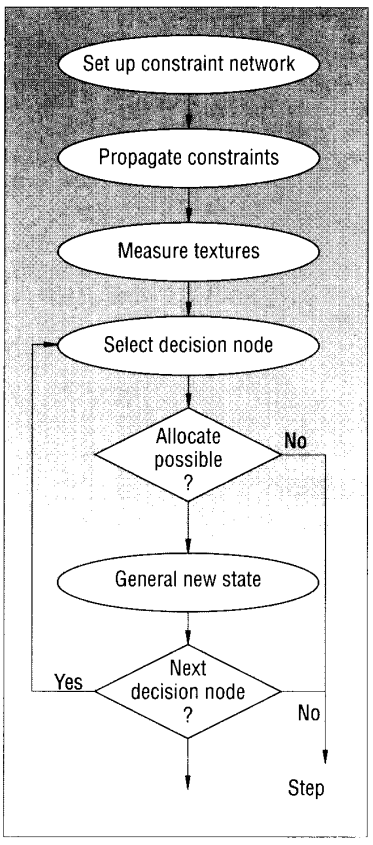


Figure 5. Algorithm for resource allocator.

to maintain the following component composition:

- Projectiles: 100 percent
- PD fuses: 60 percent
- MTSQ fuses: 60 percent

The configurator generates two feasible configuration types for each round: CT_1 and CT_2 for $Round_1$, and CT_{n-1} and CT_n for $Round_n$. Each configuration type contains feasible configurations based on compatibility constraints. The constraint graph for this problem is shown in Figure 4.

The solution process. The allocator first constructs the initial state using the constraint graph as the state's problem topology (see Figure 5). The allocator then performs the following actions:

(1) Propagate constraints.
   - Calculate component demand. Calculate each configuration's demand for a component by multiplying the expected quantity by the number of components required per configuration. For each component, determine the demand by summing each configuration's demand for the component.
   - Calculate max_rounds. For each configuration, determine how many rounds of this configuration can be created given current inventories and independent of other configurations.

(2) Measure textures.
   - Calculate component contention. For each component, determine the amount of contention by dividing demand for each component by available quantity. The greater the number, the greater the contention.
   - Calculate configuration reliance. A product's reliance on a specific configuration is a function of the number of feasible configurations of that product.

(3) Select a decision node by choosing a component and a configuration of which it is a member. For example, we can start with the most or the least contentious components, or with the configuration that requires the most components. The heuristics for selecting a decision node include:
   - Select the component with the greatest contention.
   - If there is more than one component with the same contention, select the one with the smaller number of configurations.
   - For a given component, select the configuration with the greatest reliance.

(4) Generate a new state. Once the allocator has selected a component, it generates a new state and allocates the quantity to the chosen configuration. Coral changes the constraint graph in the new state as follows:
   - Decrement the component node's quantity by the amount allocated.

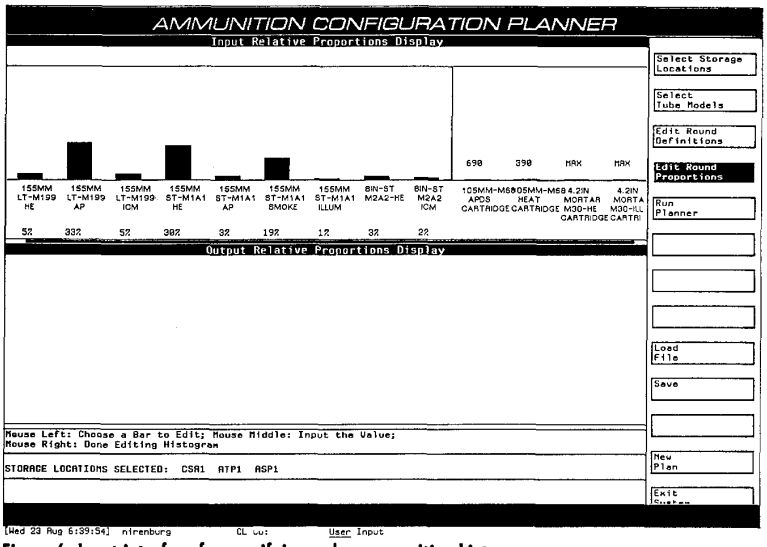


Figure 6. Input interface for specifying order-composition histograms.

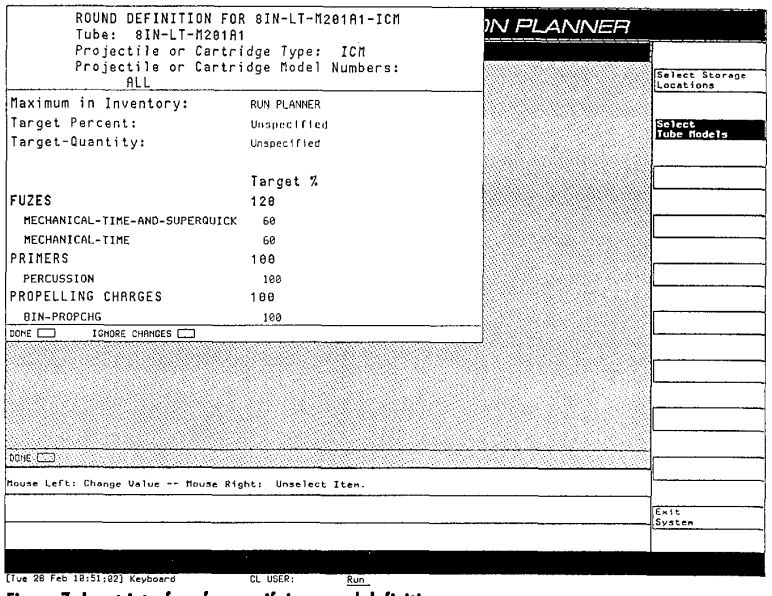


Figure 7. Input interface for specifying round definitions.

- Append the component node's allocation in the form (<configuration>, <amount allocated>).
- Set the configuration node's allocated amount to the specified amount.
- Assign and allocate other components to the chosen configuration node.

The analysis module. After running the resource allocator, Coral collects data for analysis and presentation to the user. The analysis module reports the results of the planning process, analyzes and reports on bottleneck components and remaining inventory, and replans through selective violation of constraints.

There are cases where none of the alternative configurations can be used. This means that either the user requirements have overconstrained the problem so that no solution is possible (in which case we relax the constraints), or bad choices have been made earlier in the solution process. For this reason, the system must record each decision and its alternatives so that dependency-directed backtracking can take place. The user can then answer "what-if" questions by reexecuting the algorithm after changing constraint graph parameters.

An example

We developed a histogram-based input interface to enter order-composition specifications (see Figure 6), and used a tabular report (see Figure7) to enter definitions of ammunition rounds.

Configuration. Figure 8 presents the configuration knowledge tree for the ammunition problem. Since a specification for separate loading rounds requires different ammunition components such as projectiles, propelling charges, primers, and fuses, we formed an And-node schema called Separate-loading-round and included all its parts. To include the different types of projectiles, we formed an Or-node schema called Projectile and listed alternatives such as HE, Hera, and Illum types.

In this problem, the constraints almost exclusively deal with the compatibility of components and characteristics. The most important sources of constraints are the existing compatibility charts that list binary or higher order compatibility constraints. For example, PD fuse model number MK27 is Compatabile-only-with model MK11 or M3A1 of HE projectile, if model M1 or M2 of a 40-mm artillery cannon is used.

User preferences on round type, component type, or component model provide another source of constraints. For example, at the level of rounds, the user might specify all 155-mm HE rounds (leading to many feasible configurations). At the component type level, the user might specify only PD or Prox fuses in 155-mm HE rounds. At the component model level, the user might specify only model M107 projectiles in 155-mm HE rounds (leading to the fewest feasible configurations).

Example operators include the component-round local specialist (a heuristic for obtaining an optimal sequence of component classes for a given class of round) and the component-class local specialist (a

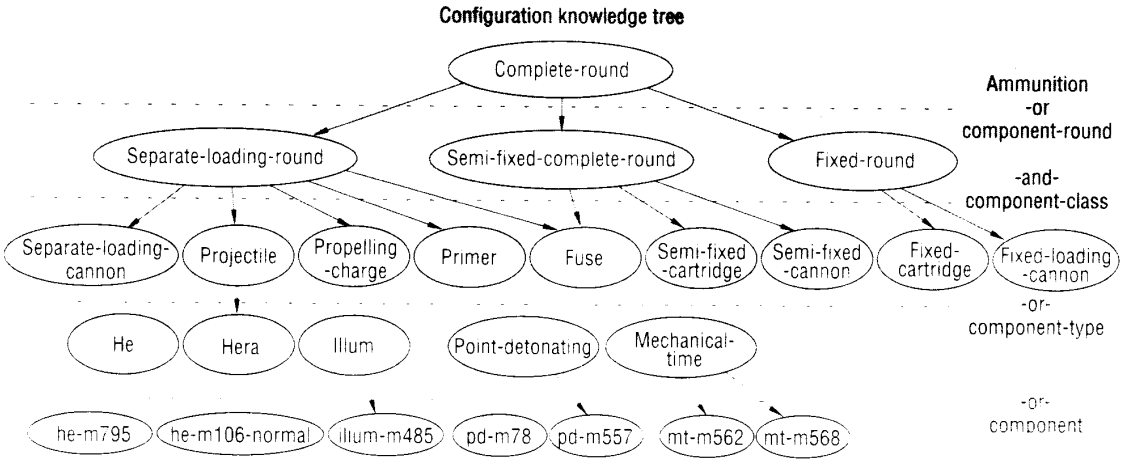


Figure 8. Disjunctive constraint graph for complete ammunition rounds.

heuristic for obtaining a reduced set of components, or model numbers, to match).

The configuration process traverses the disjunctive constraint graph, in a best-first manner from top to bottom, using operators to order the instantiation of disjuncts and conjuncts. Only selected nodes have their constraints checked.

Resource allocation. Each acceptable configuration of a complete round requires the conjunctive allocation of one or more projectiles, fuses, primers, and propellants. Specifications of the rounds define the demand for each component, which defines the demand for each component type, which defines the demand for each configuration, which defines the demand for each component resource.

Different types of constraints apply during resource allocation. The availability of the total number of components at given storage areas can affect the total number of rounds that can be configured from those locations.

Also, the user can specify preferences for configurations or compositions with different levels of detail. The user can also specify an absolute number, a relative percentage, or an objective, like "maximize." At the level of rounds, the user might specify 100 155-mm HE rounds. At the component type level, the user might also specify 60 percent of PD fuses and 60 percent of Prox fuses per round (that is, 1.2 fuses per other components).

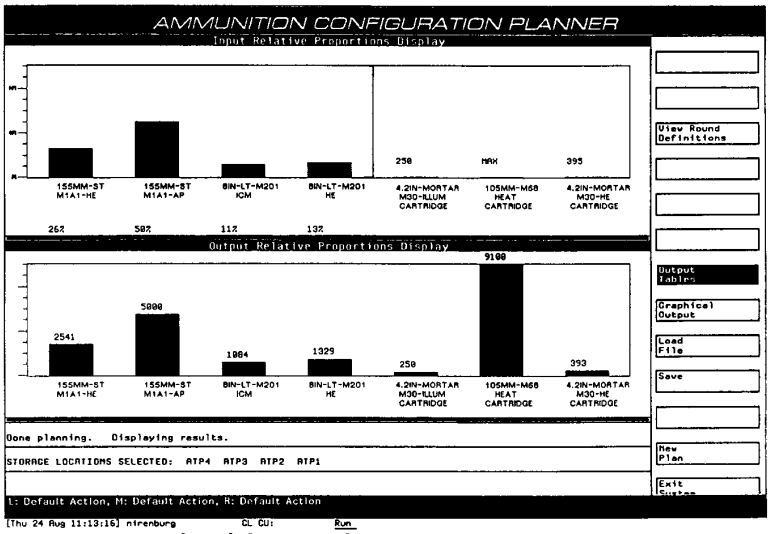


Figure 9. Output interface of planning results.

We constructed a linear-programming model of the allocation portion of this problem. Tests demonstrated that optimal solutions could be obtained in those cases where a solution existed. But the nature of the ammunition configuration problem is such that a problem specification often does not entail a feasible solution. The linear-programming approach fails to provide a solution when the component inventory cannot satisfy all the orders. To make the allocator "user-friendly," it relaxes less important constraints or user preferences to compute the solution that matches the user's needs most closely.

Figure 9 shows an example system output of the maximum number of configurable ammunition rounds. It shows that Coral has met percentage, quantity, and maximization goals effectively.

Analysis. Coral performed the analysis tasks mentioned earlier: reporting the results of the planning process, analyzing and reporting on bottleneck components and remaining inventory, and replanning

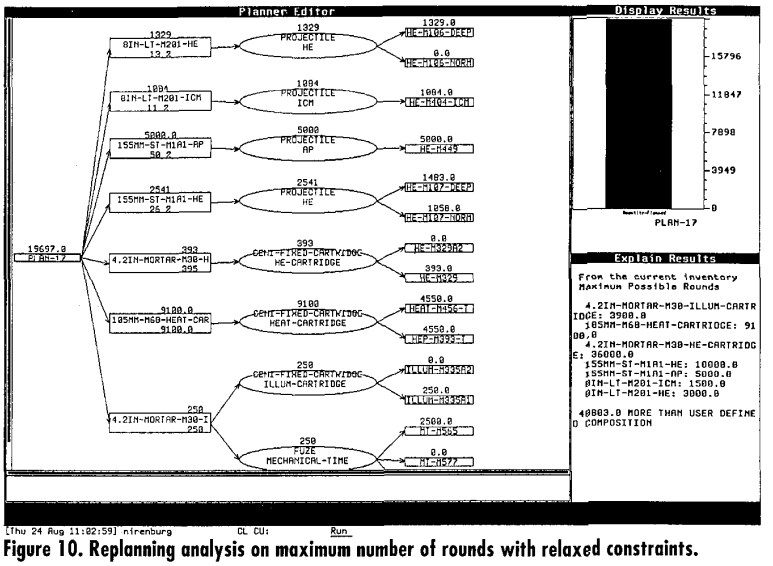


Figure 10. Replanning analysis on maximum number of rounds with relaxed constraints.

Table 1. System performance.

Test	Size	Rounds	Configurations	Constraints	CPU time (secs.)	
					Configurator	Resource allocator
1	15,000	2	84	508	9.14	3.4
2	60,000	3	100	590	16.85	8.6
3	200,000	8	198	1,026	16.31	11.3

through selective constraint violation. Figure 10 shows a replanning analysis, including how many more rounds can be configured by selectively violating unimportant constraints.

System performance. We developed three test cases with increasing complexity to validate system performance, and ran them on a Symbolics 3640. The first test case specified two rounds. The objective was to maximize the number of rounds while keeping the percentage relationship between the two rounds constant. The second test case specified three rounds, and its objective was to maximize the number of rounds. The third test case specified eight rounds: Six rounds had a percentage relationship with each other, and two rounds had a specified quantity. The objective was to maximize the number of rounds while meeting these constraints.

Table 1 presents the performance of Coral's configurator and resource allocator modules in these test cases. In the third test, with a search space of 200,000 items and more than 1,000 constraints, the system needed less than 30 seconds of processor time to generate all feasible configurations, as well as allocations of components to feasible round configurations, for given ammunition round descriptions. The overall statistics show that processor time does not relate monotonically to the number of configurations, but instead depends on the number of constraints and the complexity of underlying compatibility constraints.

CORAL DEMONSTRATES THE power of constrained heuristic search to solve configuration and allocation problems. Carnegie Group has created a generic version of the configurator, called Spex. It is a reusable shell for developing applications that access parts databases for rapid product configuration. Spex has been applied to domains such as motor configuration, thus demonstrating the generality of our approach.[10]

Research on the constrained heuristic search problem-solving paradigm continues. Major areas of concern include the development of additional textures, automating the process of problem reformulation through topological transformation, and distributed constrained heuristic search. Concurrently, we have successfully applied the paradigm to spatial planning, transportation planning and scheduling, and factory scheduling.

Acknowledgments

The Human Engineering Laboratory provided funding for case study development. We thank Philip Hayes, Peggy Andersen, and Irene Nirenburg for designing and implementing the Coral interface,[11] Thomas Chen for helping with implementation, and Florence Rouzier for editing this article before submission.

References

1. M.S. Fox, N. Sadeh, and C. Baycan, "Constrained Heuristic Search," *Proc. 11th Int'l Joint Conf. Artificial Intelligence (IJCAI-89)*, Morgan Kaufmann, San Mateo, Calif., 1989, pp. 309-316.

2. C. Dunmire et al., "Ammunition Inventory Planning: An Integration of Configuration and Resource Allocation Techniques," *Proc. Logistics Directorate (J-4) Joint Staff. Symp. on Artificial Intelligence Applications for Military Logistics*, Am. Defense Preparedness Assoc., Arlington, Va., 1990, pp. 183-191.

3. N. Sadeh and M.S. Fox, "Preference Propagation in Temporal Constraint Graphs," Tech. Report CMU-RI-TR-89-2, Robotics Inst., Carnegie Mellon Univ., Pittsburgh, 1989.

4. A.S. Kott and J.H. May, "Decomposition Versus Transformation: Case Studies of Two Models of the Design Process," *Proc. 1989 ASME Computers in Eng. Conf.*, Assoc. of Mechanical Engineers, New York, Vol. 1, 1989, pp. 1-8.

5. H.A. Simon, *The Sciences of the Artificial*, MIT Press, Cambridge, Mass., 1969.

6. M.L. Manheim, *Hierarchical Structure: A Model of Design and Planning Processes*, MIT Press, Cambridge, Mass., 1966.

7. C.A. Baykan and M.S. Fox, "Constraint Satisfaction Techniques for Spatial Planning," *Intelligent CAD Systems III:*

Practical Experience and Evaluation, P.J.W. ten Hagen and P.J. Veerkamp, eds., Springer-Verlag, Berlin, 1991, pp.187-204.

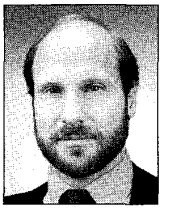
8. S. Mittal and F. Frayman, "Towards a Generic Model of Configuration Tasks," *Proc. 11th Int'l Joint Conf. Artificial Intelligence (IJCAI-89)*, Morgan Kaufmann, San Mateo, Calif., 1989, pp. 1,395-1,401.

9. S. Marcus, J. Stout, and J. McDermott, "VT: An Expert Elevator Designer That Uses Knowledge-Based Backtracking," *AI Magazine*, Vol. 9, No. 1, Spring 1988, pp. 95-114.

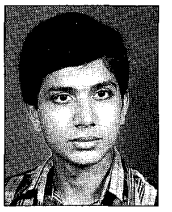
10. *Spex Knowledge-Engineering Manual, Version 1.1*, Carnegie Group, Pittsburgh, 1990.

11. P. Andersen, I. Nirenburg, and P. Hayes, "Building User Interfaces with Reusable Technology," *Knowledge Forum*, Vol. 3, No. 3, 1989, pp. 1-24.

Neena Sathi manages the cooperative processing group at Andersen Consulting's Denver office. Before joining Andersen Consulting, she managed design, development, and deployment of knowledge-based systems for telecommunication and government areas at Carnegie Group. She received her MBA from the University of Pittsburgh and her MS in chemical engineering from the University of Illinois.



Mark S. Fox is professor of industrial engineering, computer science, and management science at the University of Toronto. He received his BSc from the University of Toronto and his PhD from Carnegie Mellon University, both in computer science. In 1984 he cofounded Carnegie Group. His research interests include enterprise integration, constraint-directed reasoning, and applications of artificial intelligence to engineering and manufacturing problems. He is a fellow of AAAI and the Canadian Institute for Advanced Research, an AAAI councilor, and a member of ACM, IEEE, the Society of Mechanical Engineers, the Canadian Society for Computational Studies of Intelligence, and The Institute of Management Sciences.



Rajay Goyal is a senior knowledge engineer and a project leader at the Artificial Intelligence Technology Center at Digital Equipment Corp., where he is developing configuration solutions. Previously, he worked for Carnegie Group, where he helped design and develop the logistics planning shell. He received his MS at Louisiana State University and did graduate work in AI at Carnegie Mellon University. He is a member of AAAI.

Alexander S. Kott is a principal engineer in Carnegie Group's design and configuration group, where he specializes in applying knowledge-based techniques to automated design and configuration. He received his PhD from the University of Pittsburgh.

Readers can reach the authors in care of Mark S. Fox, Dept. of Industrial Engineering, Univ. of Toronto, 4 Taddle Creek Road, Toronto, Ontario M5S 1A4, Canada, or by e-mail at msf@ie.utoronto.ca

IEEE COMPUTER SOCIETY PRESS TITLES

NETWORKS

X.25 AND RELATED PROTOCOLS

by Uyless Black

This monograph presents a tutorial view of X.25, discusses other protocols with which it operates, and provides a convenient reference guide to its protocols. The text contains all original material, including six appendices, over 100 illustrations, and more than 50 tables.

X.25 and Related Protocols explains X.25 operations, the advantages and disadvantages of its use, the concepts and terms of packet networks, and the role other standards play in the operation of X.25. It presents a considerable amount of detailed information about X.25 and its role in various systems such as LANs, PBXs, and ISDNs.

The book covers a wide variety of subjects such as switching and routing in networks, the OSI model, physical-layer protocols and interfaces, high-level data-link control (HDLC), X.25 packet structures and types, and internetworking with SNA, DECnet, X.75, LANs, and ISDN

304 PAGES. JULY 1991. HARDBOUND. ISBN 0-8186-8976-5.
CATALOG # 1976 — $70.00 MEMBERS $45.00

BROADBAND SWITCHING:

Architectures, Protocols, Design, and Analysis

edited by Chris Dhas, Vijaya K. Konangi, and M. Sreetharan

This tutorial investigates the latest information and research on broadband switching and provides supporting material and insight into the correlated areas of networking, performance analysis, and alternate technologies. The text describes broadband switching architectures, performance modeling techniques, multistage interconnection networks, architectural options available for switches, and experimental architectures for ISDN and ATM techniques

Broadband Switching: Architectures, Protocols, Design, and Analysis also examines numerous trends in network architectures designed to meet the user's high bandwidth requirements, packet replication and switching in broadcast switching systems, important issues of bandwidth allocation and flow and congestion control, performance modeling, and photonic switching techniques and technology.

528 PAGES. AUGUST 1991. HARDBOUND. ISBN 0-8186-8926-9.
CATALOG # 1926 — $75.00 MEMBERS $50.00

ORDER TODAY! —

1-800-CS-BOOKS
or FAX (714) 821-4010
in California call (714) 821-8380